ASEN5317 - Computational Fluid Dynamics
Instructor: Professor L. Kantha

Homework #2:
Numerical Integration

by

Joseph P. Kubitschek
25 September 1998

ABSTRACT
Various numerical integration techniques were used to evaluate the integrals of different functions to gain familiarity with those techniques. For problem 1, the Trapezoidal Rule and Simpson's Rule were employed to develop code and evaluate two integrals, $f(x) = 1/(1+x^2)$ and $f(x) = e^x \sin(\pi x)$ over the range $0 \leq x \leq 1$, corresponding with two different step sizes, h=0.10 and 0.05. The results were compared from the standpoint of relative accuracy and indicate that Simpson's Rule provides fifth order accuracy while the Trapezoidal Rule only provides third order accuracy, as indicated by theory. Furthermore, Richardson's Extrapolation was used to improve the accuracy of the results. For problem 2, existing code was obtained from Numerical Recipes[1] and used to compute the integral of $f(x) = x^4 \log(x+(x^2+1)^{1/2})$ over the range $0 \leq x \leq 2$ using the various "refined" techniques available (i.e. QTRAP, QSIMP, QROMB, POLYINT, and TRAPZD). Finally, part 3, includes the integral evaluation of $f(x) = \sin(x)/x$ over the range $0 \leq x \leq \pi/2$. Existing code was modified to handle the singularity that exists at the lower limit, x = 0.

INTRODUCTION

The purpose of this project was to gain increased familiarity with FORTRAN coding through the integral evaluation of various functions using numerical integration techniques.

Background

Integration, in it simplest sense, consists of evaluating the area under a curve, $y = f(x)$ over a specified range. Theoretically, integration of some function $f(x)$ is equivalent to solving the differential equation $dy/dx = f(x)$. In this case, the limits of integration are merely the required boundary conditions. It was recognized early in history that such problems could be solved using various numerical integration techniques. Such techniques, also called quadrature, became very powerful with the advent of the computer. As such, the various techniques have been refined to improve accuracy and efficiency in the context of computing. Of the many techniques developed, this investigation focuses on three, namely the Trapezoidal Rule, Simpson's Rule, and the Romberg Method.

The Trapezoidal Rule is a two-point formula that is exact for polynomials of order 1 or less (i.e. a straight lines). The formula is obtained by passing a straight line through two points and then evaluating the area under this line (*Kantha*, *Lecture Notes*). Using Lagrange interpolation, the formula is obtained as

$$\int f(x)dx = h[f_1 + f_2]/2 + O(h^3 f''),$$

where, h = interval or step size.

The resulting Trapezoidal Rule formula is accurate to order 3, a result of the truncation error associated with neglecting higher order terms. This formula may be extended to a specified degree of accuracy by dividing the range of integration into (n-1) intervals. The resulting formula is

$$\int f(x)dx = h[f_1/2 + f_2 + \ldots + f_{n-1} + f_n/2] + O(1/n^2).$$

Alternatively, Simpson's Rule is a three-point formula that is exact for polynomials of order three or less. The primary advantage of this formula over the Trapezoidal Rule formula is improved accuracy for higher order polynomials (*Kantha*, *Lecture Notes*). The resulting formula obtained by passing a second order polynomial through three points is

$$\int f(x)dx = h[f_1 + 4f_2 + f_3]/3 + O(h^5 f^{(4)}).$$

This formula is accurate to order 5 and may also be extended to improve accuracy to a specified degree by interval halving. Again, this is achieved by dividing the range of integration into (n-1) intervals. The resulting formula is

$$\int f(x)dx = h[f_1 + 4f_2 + 2f_3 + 4f_4 + \ldots + 2f_{n-2} + 4f_{n-1} + f_n]/3 + O(1/n^4).$$

For both techniques previously described, a technique known as Richardson extrapolation may be employed to improve accuracy. The last technique considered here is called the Romberg Method and uses this extrapolation technique to improve accuracy of the simple Trapezoidal Rule.

Solution Technique

For problem 1, the evaluation of the integrals of $f(x) = 1/(1+x^2)$ and $f(x) = e^x\sin(\pi x)$ over the range $0 \leq x \leq 1$, code was developed independently using the Trapezoidal Rule and Simpson's Rule. These programs, called *trap.f*, *trap2.f*, *simp.f*, and *simp2.f* divide the range of integration into equal intervals, h = 0.10 and h = 0.05 corresponding with 10 and 20 equal intervals, respectively and compute the values of the integrals over the specified range. The relative error is then computed using the exact values. Finally, Richardson extrapolation is used to improve accuracy. In each case, the relative error is compared. The exact value was obtained using the CRC Standard Mathematical Tables and Formulae[5], table of integrals. The theoretically exact results were obtained as

$$\int 1/(1+x^2)dx = \tan^{-1}(x),$$

$$\int e^x\sin(\pi x)dx = e^x[\sin(\pi x) - \pi\cos(\pi x)]/(1+\pi^2),$$

from which the exact values were computed over the limits of integration, $0 \leq x \leq 1$.

Problem 2 also consists of numerical integration using the described techniques. However, existing "refined" code, obtained from Numerical Recipes[1], is used to evaluate the integral of $f(x) = x^4\log(x+(x^2+1)^{1/2})$ over the range $0 \leq x \leq 2$. For part a, QTRAP and TRAPZD are used in the driver code *hw22a.f*; For part b, QSIMP and TRAPZD are used in the driver code *hw22b.f*; And, for part c, QROMB, POLYINT, and TRAPZD are used in the driver code *hw22c.f*. The exact value was obtained from the table of integrals as

$$\int x^4\log(x+(x^2+1)^{1/2})dx = (x^5/5)\log[x+(x^2+1)^{1/2}] - (x^2+1)^{1/2}[(x^4/25)+(4x^2/75)-(8/75)]$$

Finally, for problem 3, TRAPZD is replaced by an open formula to evaluate the integral of $f(x) = \sin(x)/x$ over the range $0 \leq x \leq \pi/2$. This routine makes use of MIDPNT, in the driver code *hw23.f*, to handle the singularity at the lower limit of integration, x = 0. The exact value was obtained as before as

$$\int [\sin(x)/x]dx = \Sigma_0^\infty\ (-1)^n(x)^{2n+1}/(2n+1)(2n+1)! \quad \text{(infinite series)}$$

RESULTS AND DISCUSSION

Table 1 represents the results of problem 1. Each of the integrals were evaluated using the methods described and the results were tabulated accordingly. The results indicate the relative degree of accuracy inherent in each numerical integration technique. Simpson's Rule in both integral evaluation cases reflects improved accuracy over the Trapezoidal Rule as expected. Additionally, one can see the improvement in accuracy through the use of Richardson extrapolation. For the Trapezoidal Rule, Richardson extrapolation improves the degree of accuracy by a minimum of two orders of magnitude.

Table 1. - Numerical integration results (Problem 1a.-d.)

	Trapezoidal Rule		**Simpson's Rule**	
f(x) =	***$1/(1+x^2)$***	***$\exp(x)\sin(\pi x)$***	***$1/(1+x^2)$***	***$\exp(x)\sin(\pi x)$***
Exact	0.785398	1.074678	0.78539816	1.07467819
Comp h = 0.10	0.784982	1.064967	0.78539824	1.07472277
Error	**0.000416**	**0.009711**	**-0.00000008**	**0.00004458**
Comp h = 0.05	0.785294	1.072278	0.78539813	1.07476400
Error	**0.000104**	**0.002400**	**0.00000004**	**0.00000359**
Richardson Extrap.	0.785398	1.074715	0.78539814	1.07476675
Error	**0.000000**	**-0.000037**	**0.00000002**	**0.00000084**

Note: Error = Exact - Computed.

For the refined subroutines used in problem 2, the relative accuracy of each numerical integration technique is also visible. Table 2 represents the results of each technique. It is interesting to note that all three techniques return the same value to the specified degree of accuracy as guaranteed by the refinement that is inherent. An attempt was made to assess the efficiency of each algorithym using a call to the CPU clock. However, this approach was not successful as the command used did not return a result. Alternatively, a count of the number of iterations or function evaluations could have been used to compare relative efficiency.

Table 2. - Numerical integration results using "refined" subroutines (Problem 2a.-c.)

	QTRAP	**QSIMP**	**QROMB**
Computed Results	3.540961	3.540961	3.540961

Finally, Table 3 represents the results of the integral value of f(x) = sin(x)/x. In this case an open formula was used to handle the singularity which exists at the lower limit, x = 0.

Table 3. - Numerical integration results (Problem 3.)

	MIDPNT
Computed Result	1.370615
Exact	1.370762
Error	**0.000146**

Note: Error = Exact - Computed.

CONCLUSIONS

- Simpson's Rule provides improved accuracy over the Trapezoidal Rule consistent with theory and as demonstrated by the results from problem 1.

- Richardson extrapolation is a powerful technique for improving accuracy of both the Trapezoidal Rule and Simpson's Rule. In the case of the Trapezoidal Rule, Richardson extrapolation improves accuracy by at least two orders of magnitude.

- The refined techniques obtained from Numerical Recipes[1] are powerful and efficient methods of integral evaluation to a degree of accuracy specified *apriori*.

- Singularities that may exist can easily be handled using refined techniques also obtained from Numerical Recipes[1]. In this case, open formulas are used.

REFERENCES

1. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., Numerical Recipes in Fortran 77, 2nd Edition, Volume 1, Cambridge University Press, 1992.

2. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., Numerical Recipes in Fortran 77 – Examples Book, Cambridge University Press, 1992.

3. Chapra, S.C., and Canale, R.C., Introduction to Computing for Engineers, McGraw-Hill Book Co., 1986.

4. Loukides, M., UNIX for Fortran Programmers, O'Reilly and Associates, Inc., 1990.

5. Zwillinger, D., CRC Standard Mathematical Tables and Formulae, 30th Edition, CRC Press, 1996.

APPENDIX A – FORTRAN CODE

PROBLEM 1.a.)

```
c      Program trap.f
c
c      ASEN5317 - Assign#2
c      J.Kubitschek 9/18/98
c
c      Program evaluates integral of two functions:
c        1.)  f(x)= 1/(1+x^2)
c        2.)  f(x)= exp(x)sin(pi*x)
c      over the range 0<=x<=1 using trapezoidal rule.  Range is
c      divided into 10 equal intervals.  Results include integral
c      values and global error.
       INTEGER n
       REAL x,pi,del,s1,s2,f1,f2,e1,e2,r1,r2,f1sum,f2sum
c
       pi=3.14159265
       x=0.0
       f1sum=1.5
       f2sum=exp(1.0)*sin(pi)
       s1=0.0
       s2=0.0
       e1=0.0
       e2=0.0
       r1=0.0
       r2=0.0
       del=1.0/10.0
       do n=1,9
         x=n*del
         f1=2.0/(1.0+x**2)
         f2=2.0*exp(x)*sin(pi*x)
         s1=s1+f1
         s2=s2+f2
       end do
       r1=(del/2.0)*(s1+f1sum)
       r2=(del/2.0)*(s2+f2sum)
       e1=atan(1.0)-r1
       e2=((pi/(1+pi**2))*(exp(1.0)+1))-r2
c
       open (unit=45,file='hw2_1a1.out',status='unknown')
       write (45,*)'trap.f output'
       write (45,*)'   f1(x)','    error1','    f2(x)','    error2'
       write (45,10) r1,e1,r2,e2
10     format(1x,f8.6,2x,f8.6,2x,f8.6,2x,f8.6)
       END
```

PROBLEM 1.b.)

```
c      Program trap2.f
c
c      ASEN5317 - Assign#2
c      J.Kubitschek 9/18/98
c
c      Program evaluates integral of two functions:
c        1.)  f(x)= 1/(1+x^2)
c        2.)  f(x)= exp(x)sin(pi*x)
c      over the range 0<=x<=1 using trapezoidal rule.  Range is
c      divided into 20 equal intervals.  Results include integral
c      values and global error.
      INTEGER n
      REAL x,pi,del,s1,s2,f1,f2,e1,e2,r1,r2,f1sum,f2sum
c
      pi=3.14159265
      x=0.0
      f1sum=1.5
      f2sum=exp(1.0)*sin(pi)
      s1=0.0
      s2=0.0
      e1=0.0
      e2=0.0
      r1=0.0
      r2=0.0
      del=1.0/20.0
      do n=1,19
        x=n*del
        f1=2.0/(1.0+x**2)
        f2=2.0*exp(x)*sin(pi*x)
        s1=s1+f1
        s2=s2+f2
      end do
      r1=(del/2.0)*(s1+f1sum)
      r2=(del/2.0)*(s2+f2sum)
      e1=atan(1.0)-r1
      e2=((pi/(1+pi**2))*(exp(1.0)+1))-r2
c
      open (unit=45,file='hw2_1b1.out',status='unknown')
      write (45,*)'trap2.f output'
      write (45,*)'   f1(x)','    error1','     f2(x)','    error2'
      write (45,10) r1,e1,r2,e2
10    format(1x,f8.6,2x,f8.6,2x,f8.6,2x,f8.6)
      END
```

PROBLEM 1.a.)

```
c       Program simp.f
c
c       ASEN5317 - Assign#2
c       J.Kubitschek 9/18/98
c
c       Program evaluates integral of two functions:
c         1.)  f(x)= 1/(1+x^2)
c         2.)  f(x)= exp(x)sin(pi*x)
c       over the range 0<=x<=1 using Simpson's rule.  Range is
c       divided into 10 equal intervals.  Results include integral
c       values and global error.
        INTEGER n
        REAL x,pi,del,s1,s2,f1,f2,e1,e2,r1,r2,f1sum,f2sum
c
        pi=3.141592654
        x=0.0
        f1sum=1.5
        f2sum=exp(1.0)*sin(pi)
        s1=0.0
        s2=0.0
        e1=0.0
        e2=0.0
        r1=0.0
        r2=0.0
        del=1.0/10.0
        do n=1,9,2
          x=n*del
          f1=4.0/(1.0+x**2)
          f2=4.0*exp(x)*sin(pi*x)
          s1=s1+f1
          s2=s2+f2
        end do
        do n=2,8,2
          x=n*del
          f1=4.0/(1.0+x**2)
          f2=4.0*exp(x)*sin(pi*x)
          s1=s1+f1
          s2=s2+f2
        end do
        r1=(del/3.0)*(s1+f1sum)
        r2=(del/3.0)*(s2+f2sum)
        e1=atan(1.0)-r1
        e2=((pi/(1+pi**2))*(exp(1.0)+1))-r2
c
        open (unit=45,file='hw2_1a2.out',status='unknown')
        write (45,*)'simp.f output'
        write (45,*)'   f1(x)','    error1','    f2(x)','    error2'
        write (45,10) r1,e1,r2,e2
10      format(1x,f10.8,2x,f10.8,2x,f10.8,2x,f10.8)
        END
```

PROBLEM 1.b.)

```
c       Program simp2.f
c
c       ASEN5317 - Assign#2
c       J.Kubitschek 9/18/98
c
c       Program evaluates integral of two functions:
c         1.)  f(x)= 1/(1+x^2)
c         2.)  f(x)= exp(x)sin(pi*x)
c       over the range 0<=x<=1 using Simpson's rule.  Range is
c       divided into 20 equal intervals.  Results include integral
c       values and global error.
        INTEGER n
        REAL x,pi,del,s1,s2,f1,f2,e1,e2,r1,r2,f1sum,f2sum
c
        pi=3.141592654
        x=0.0
        f1sum=1.5
        f2sum=exp(1.0)*sin(pi)
        s1=0.0
        s2=0.0
        e1=0.0
        e2=0.0
        r1=0.0
        r2=0.0
        del=1.0/20.0
        do n=1,19,2
          x=n*del
          f1=4.0/(1.0+x**2)
          f2=4.0*exp(x)*sin(pi*x)
          s1=s1+f1
          s2=s2+f2
        end do
        do n=2,18,2
          x=n*del
          f1=2.0/(1.0+x**2)
          f2=2.0*exp(x)*sin(pi*x)
          s1=s1+f1
          s2=s2+f2
        end do
        r1=(del/3.0)*(s1+f1sum)
        r2=(del/3.0)*(s2+f2sum)
        e1=atan(1.0)-r1
        e2=((pi/(1+pi**2))*(exp(1.0)+1))-r2
c
        open (unit=45,file='hw2_1b2.out',status='unknown')
        write (45,*)'simp2.f output'
        write (45,*)'   f1(x)','    error1','    f2(x)','    error2'
        write (45,10)r1,e1,r2,e2
10      format(1x,f10.8,2x,f10.8,2x,f10.8,2x,f10.8)
        END
```

PROBLEM 2 CODE:

Driver Code:

Problem 2a.) Extended Trapezoidal Rule

```
      PROGRAM hw22a
      REAL func,a,b,s
      EXTERNAL func
      a=0.0
      b=2.0
      s=0.0
C
      call qtrap(func,a,b,s)
C
      open(unit=45,file='hw22a.out',status='unknown')
      write(45,*)'hw2pr2a integral value = '
      write(45,10) s
10    format(2x,f10.8)
      close(45)
      END
C
      REAL FUNCTION func(x)
      REAL x
      func=(x**4)*alog10(x+sqrt(x**2+1))
      END
```

Problem 2b.) Simpson's Rule

```
      PROGRAM hw22b
      REAL func,a,b,s
      EXTERNAL func
      a=0.0
      b=2.0
      s=0.0
C
      call qsimp(func,a,b,s)
C
      open(unit=45,file='hw22b.out',status='unknown')
      write(45,*)'hw2pr2b integral value = '
      write(45,10) s
10    format(2x,f10.8)
      close(45)
      END
C
      REAL FUNCTION func(x)
      REAL x
      func=(x**4)*alog10(x+sqrt(x**2+1))
      END
```

Problem 2c.) Romberg Method

```
      PROGRAM hw22c
      REAL func,a,b,s
      EXTERNAL func
      a=0.0
      b=2.0
      s=0.0
C
      call qromb(func,a,b,s)
C
      open(unit=45,file='hw22c.out',status='unknown')
      write(45,*)'hw2pr2c integral value = '
      write(45,10) s
10    format(2x,f10.8)
      close(45)
      END
C
      REAL FUNCTION func(x)
      REAL x
      func=(x**4)*alog10(x+sqrt(x**2+1))
      END
```

Subroutines/Modules:

```
      SUBROUTINE trapzd(func,a,b,s,n)
      INTEGER n
      REAL a,b,s,func
      EXTERNAL func
      INTEGER it,j
      REAL del,sum,tnm,x
      if (n.eq.1) then
        s=0.5*(b-a)*(func(a)+func(b))
      else
        it=2**(n-2)
        tnm=it
        del=(b-a)/tnm
        x=a+0.5*del
        sum=0.
        do 11 j=1,it
          sum=sum+func(x)
          x=x+del
11        continue
        s=0.5*(s+(b-a)*sum/tnm)
      endif
      return
      END

      SUBROUTINE qtrap(func,a,b,s)
      INTEGER JMAX
      REAL a,b,func,s,EPS
      EXTERNAL func
      PARAMETER (EPS=1.e-6, JMAX=20)
CU    USES trapzd
      INTEGER j
      REAL olds
      olds=-1.e30
      do 11 j=1,JMAX
        call trapzd(func,a,b,s,j)
        if (abs(s-olds).lt.EPS*abs(olds)) return
        olds=s
11    continue
      pause 'too many steps in qtrap'
      END

      SUBROUTINE qsimp(func,a,b,s)
      INTEGER JMAX
      REAL a,b,func,s,EPS
      EXTERNAL func
      PARAMETER (EPS=1.e-6, JMAX=20)
CU    USES trapzd
      INTEGER j
      REAL os,ost,st
      ost=-1.e30
      os= -1.e30
      do 11 j=1,JMAX
```

```
        call trapzd(func,a,b,st,j)
        s=(4.*st-ost)/3.
        if (abs(s-os).lt.EPS*abs(os)) return
        os=s
        ost=st
11    continue
      pause 'too many steps in qsimp'
      END

      SUBROUTINE qromb(func,a,b,ss)
      INTEGER JMAX,JMAXP,K,KM
      REAL a,b,func,ss,EPS
      EXTERNAL func
      PARAMETER (EPS=1.e-6, JMAX=20, JMAXP=JMAX+1, K=5, KM=K-1)
CU    USES polint,trapzd
      INTEGER j
      REAL dss,h(JMAXP),s(JMAXP)
      h(1)=1.
      do 11 j=1,JMAX
        call trapzd(func,a,b,s(j),j)
        if (j.ge.K) then
          call polint(h(j-KM),s(j-KM),K,0.,ss,dss)
          if (abs(dss).le.EPS*abs(ss)) return
        endif
        s(j+1)=s(j)
        h(j+1)=0.25*h(j)
11    continue
      pause 'too many steps in qromb'
      END

      SUBROUTINE polint(xa,ya,n,x,y,dy)
      INTEGER n,NMAX
      REAL dy,x,y,xa(n),ya(n)
      PARAMETER (NMAX=10)
      INTEGER i,m,ns
      REAL den,dif,dift,ho,hp,w,c(NMAX),d(NMAX)
      ns=1
      dif=abs(x-xa(1))
      do 11 i=1,n
        dift=abs(x-xa(i))
        if (dift.lt.dif) then
          ns=i
          dif=dift
        endif
        c(i)=ya(i)
        d(i)=ya(i)
11    continue
      y=ya(ns)
      ns=ns-1
      do 13 m=1,n-1
        do 12 i=1,n-m
          ho=xa(i)-x
          hp=xa(i+m)-x
```

```
          w=c(i+1)-d(i)
          den=ho-hp
          if(den.eq.0.)pause 'failure in polint'
          den=w/den
          d(i)=hp*den
          c(i)=ho*den
12        continue
        if (2*ns.lt.n-m)then
          dy=c(ns+1)
        else
          dy=d(ns)
          ns=ns-1
        endif
        y=y+dy
13      continue
      return
      END
```

PROBLEM 3 - REVISED CODE

```
      PROGRAM hw23
      INTEGER NMAX
      PARAMETER(NMAX=10)
      INTEGER i
      REAL func,a,b,s,pi
      EXTERNAL func
      pi=3.14159265
      a=0.0
      b=pi/2.0
      s=0.0
      do 11 i=1,NMAX
        call midpnt(func,a,b,s,i)
11    continue
      open(unit=45,file='hw23.out',status='unknown')
      write(45,*)'hw23 integral value = '
      write(45,10) s
10    format(2x,f12.8)
      close(45)
      END
C
      REAL FUNCTION func(x)
      REAL x
      func=sin(x)/x
      END

      SUBROUTINE midpnt(func,a,b,s,n)
      INTEGER n
      REAL a,b,s,func
      EXTERNAL func
      INTEGER it,j
      REAL ddel,del,sum,tnm,x
      if (n.eq.1) then
        s=(b-a)*func(0.5*(a+b))
      else
        it=3**(n-2)
        tnm=it
        del=(b-a)/(3.*tnm)
        ddel=del+del
        x=a+0.5*del
        sum=0.
        do 11 j=1,it
          sum=sum+func(x)
          x=x+ddel
          sum=sum+func(x)
          x=x+del
11        continue
        s=(s+(a-b)*sum/tnm)/3.
      endif
Return
END
```

APPENDIX B – OUTPUT

Problem 1.a.)

trap.f output

	f1(x)	error1	f2(x)	error2
Computed	.784982	.000416	1.064967	.009711
Exact	.785398		1.074678	

Problem 1.b.)

trap2.f output

	f1(x)	error1	f2(x)	error2
Computed	.785294	.000104	1.072278	.002400
Exact	.785398		1.074678	
Richardson Ext.	.785398	.000000	1.074715	-.000037

Problem 1.a.)

simp.f output

	f1(x)	error1	f2(x)	error2
Computed	.78539824	-.00000008	1.07472277	.00004458
Exact	.78539816		1.07467819	

Problem 1.b.)

simp2.f output

	f1(x)	error1	f2(x)	error2
Computed	.78539813	.00000004	1.07476400	.00000359
Exact	.78539816		1.07467819	
Richardson Ext.	.78539814	.00000002	1.07476675	.00000084

Problem 2a.)
hw2pr2a integral value = 3.540961

Problem 2b.)
hw2pr2b integral value = 3.540961

Problem 2c.)
hw2pr2c integral value = 3.540961

Problem 3.)
hw23 integral value = 1.370615